uplevel manual page - Built-In Commands

tcl.tk/man/tcl/TclCmd/uplevel.htm

NAME

uplevel — Execute a script in a different stack frame

SYNOPSIS

uplevel ?*level*? *arg* ?*arg ...*?

DESCRIPTION

All of the *arg* arguments are concatenated as if they had been passed to **concat**; the result is then evaluated in the variable context indicated by *level*. **Uplevel** returns the result of that evaluation.
If *level* is an integer then it gives a distance (up the procedure calling stack) to move before executing the command. If *level* consists of **#** followed by a number then the number gives an absolute level number. If *level* is omitted then it defaults to **1**. *Level* cannot be defaulted if the first *command* argument starts with a digit or **#**.

For example, suppose that procedure **a** was invoked from top-level, and that it called **b**, and that **b** called **c**. Suppose that **c** invokes the **uplevel** command. If *level* is **1** or **#2** or omitted, then the command will be executed in the variable context of **b**. If *level* is **2** or **#1** then the command will be executed in the variable context of **a**. If *level* is **3** or **#0** then the command will be executed at top-level (only global variables will be visible).

The **uplevel** command causes the invoking procedure to disappear from the procedure calling stack while the command is being executed. In the above example, suppose **c** invokes the command

```
uplevel 1 {set x 43; d}
```

where **d** is another Tcl procedure. The **set** command will modify the variable **x** in **b**'s context, and **d** will execute at level 3, as if called from **b**. If it in turn executes the command

```
uplevel {set x 42}
```

then the **set** command will modify the same variable **x** in **b**'s context: the procedure **c** does not appear to be on the call stack when **d** is executing. The **info level** command may be used to obtain the level of the current procedure.

Uplevel makes it possible to implement new control constructs as Tcl procedures (for example, **uplevel** could be used to implement the **while** construct as a Tcl procedure).

The **namespace eval** and **apply** commands offer other ways (besides procedure calls) that the Tcl naming context can change. They add a call frame to the stack to represent the namespace context. This means each **namespace eval** command counts as another call level for **uplevel** and **upvar** commands. For example, **info level 1** will return a list describing a command that is either the outermost procedure call or the outermost **namespace eval** command. Also, **uplevel #0** evaluates a script at top-level in the outermost namespace (the global namespace).

EXAMPLE

As stated above, the **uplevel** command is useful for creating new control constructs. This example shows how (without error handling) it can be used to create a **do** command that is the counterpart of **while** except for always performing the test after running the loop body:

```
proc do {body while condition} {
    if {$while ne "while"} {
        error "required word missing"
    }
    set conditionCmd [list expr $condition]
    while {1} {
        uplevel 1 $body
        if {![uplevel 1 $conditionCmd]} {
            break
        }
    }
}
```